FACILITY FORM 602

N71-37739
(ACCESSION NUMBER)

34
(PAGES)

CR-123214
(NASA CR OR TMX OR AD NUMBER)

(THRU)

G3
(CODE)

08
(CATEGORY)

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

Section 314
Technical Memorandum No. 240

COMPUTING QUADRATIC PROGRAMMING PROBLEMS: LINEAR INEQUALITY AND EQUALITY CONSTRAINTS

by

Richard J. Hanson

February 9, 1970

California Institute of Technology

JET PROPULSION LABORATORY
4800 Oak Grove Drive
Pasadena, California 91103

Computing Quadratic Programming Problems: Linear Inequality and Equality Constraints*

ABSTRACT

This paper presents a computational method for the general quadratic programming problem with linear (inequality and equality) constraints.

These problems can be reformulated as 'hard core' quadratic programs which involve minimizing a sum of squares of the unknown parameters subject to inequality constraints alone. This 'hard core' problem is solved and the solution can be obtained by explicit linear transformations and translations to the original coordinates.

An application of (least squares) constrained curve fitting (by general cubic splines) is presented.

*This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract NAS7-100, sponsored by the National Aeronautics and Space Administration.

1. INTRODUCTION

I will present an algorithm for solving any general quadratic programming problem with linear constraints. This includes the cases where the coefficient matrix is rank deficient. Other algorithms have been suggested by a variety of authors (Ref. [1], p. 93-174, esp. p. 171-174 and Golub and Saunders [2]). The main point of this paper is that a reformulation of the problem leads to a more numerically stable algorithm.

An application of this algorithm to curve fitting (by cubic splines) with equality and inequality constraints on the function and its derivatives is given in Section 5.

Most quadratic programs are stated in the form of

<u>Problem P</u>

<u>Let</u> A,b,C,d,E <u>and</u> f <u>respectively denote</u> $m_A \times n$, $m_A \times 1$, $m_C \times n$, $m_C \times 1$, $m_E \times n$, <u>and</u> $m_E \times 1$ <u>real matrices</u>. <u>Find</u> x <u>such that the euclidean norm</u>,

$$\|Ax-b\|, \text{ is minimized} \tag{1.1}$$

<u>subject to</u>

$$Ex = f \tag{1.2}$$

<u>and</u>

$$Cx \geq d. \tag{1.3}$$

Here the inequality of (1.3) means that all m_C inequalities are to hold simultaneously.

A reliable computational capacity for solving Problem P is extremely important when one is computing a solution to a linear system of algebraic equations where the solution variables are to have certain bounds or constraints which can be expressed in terms of linear combinations of the variables. Such applications, to name only three, occur in curve fitting, nonlinear constrained parameter estimation with Newton's method, and the computation of nonnegative approximate solutions to linear integral equations.

In Section 3 I will show that the general situation of Problem P can be reformulated as another quadratic programming problem,

<u>Problem Q</u>

<u>Find</u> $x = (x_1,\ldots,x_n)^T$ <u>such that</u>

$$f(x) = \tfrac{1}{2}\sum_{j=1}^{r} x_j^2, \; (1 \le r \le n), \text{ \underline{is minimized}} \qquad (1.4)$$

<u>subject to</u>

$$Cx \ge d. \qquad (1.5)$$

The matrix C and vector d of the inequality (1.5) are $m \times n$ and $m \times 1$ respectively. They are not generally the same as C and d of Eq. (1.3). I'll use the same symbols to ease the notation.

Finally I mention here that this algorithm for solving Problem P is most appropriate for those problems where computer core storage is available for the matrices A, C and E.

During the remainder of this paper, when convenient, I'm going to refer to other work which describes the details of the computations which will be needed.

The key procedure for solving Problem Q is the gradient projection algorithm of J. B. Rosen (see [3]). I've modified this algorithm and given a method for removing and adding columns to a matrix whose pseudoinverse (matrix giving the least squares solution of shortest length) must be computed. This method is numerically very stable.

The computational method used to solve Problem Q involves an application of the following theorem ([1] p. 114).

Theorem 1.1 (Kuhn-Tucker Conditions)

A necessary and sufficient condition that the function f of Problem Q attain its minimum at a point x with $Cx \geq d$ is that these exist an m-vector y such that

$$C^T y = [x_1,\ldots,x_r,0,\ldots,0]^T, \quad (y \geq 0).$$

Rosen's gradient projection algorithm [3], as I've modified it, finds an n-vector x and an m-vector y so that the conditions of Theorem 1.1 are satisfied. Rosen's original algorithm can be expected to work well for the case $r=n$ because the projected gradient always points toward the unconstrained minimum of $f(x)$ restricted to a linear (constraint) flat.

In case $r < n$ the projected gradient may not point toward the unconstrained minimum of $f(x)$ on this linear (constraint) flat. This unconstrained minimum can be obtained by solving a certain least squares problem with linear equality constraints. The details for this and a computational procedure for accomplishing it are described in Section 3, Procedure L3.

2. NOTATIONAL REMARKS

I'll say that a matrix A (of real numbers) is m x n if A has m rows and n columns. Whenever there may be confusion I'll write A_{mxn} to designate this.

The symbols I_p and O_p respectively stand for the p x p identity and the q x q zero matrices. By $A \oplus B$ I'll mean the direct sum of (square matrices) A and B, the block diagonal $A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$. The m x n matrix of all zeros will be designated by O_{mxn} whenever there is any question of the dimensions. The norm of a vector x, $\|x\|$, will mean the euclidean norm, $\|x\| = (x^T x)^{\frac{1}{2}}$. The corresponding matrix norm for A_{mxn} is $\|A\| = \max_{\|x\|=1} \|Ax\| = s_1$, where s_1 is the largest singular value of A. (See Golub and Reinsch [3]). An m x n matrix Q, $(m \geq n)$, is orthonormal if $Q^T Q = I_n$.

The symbol ϵ will denote the relative precision of a computer word; thus ϵ is the largest number of all positive numbers η such that $1+\eta$ equals 1 after the computer floating point addition.

3. REFORMULATING PROBLEM P AND SOLVING PROBLEM Q

The procedure of reformulating Problem P to have the form of Problem Q is a five step procedure:

STEP	
1	Remove the linear equality constraints and obtain a problem of the same general form as Problem P but with linear equality constraints not explicitly present.
2	Make linear changes of variables, and a translation of coordinates, so that Problem P reduces to the 'hard core' problem of minimizing a sum of squares subject to linear inequality constraints.
3	Solve this 'hard core' problem, Problem Q.
4	Reapply the transformations of Step 2.
5	Reapply the transformations of Step 1 to obtain the solution of Problem P as stated.

More precisely, the most general solution of Eq. (1.2) has the form $x = x_0 + Hy$ where $Ex_0 = f$ and the columns of the matrix H are an orthonormal basis for the subspace of all solutions to $Ex = 0$. Thus Eq. (1.2) is satisfied if $x = x_0 + Hy$ and the vector y (whose dimension is the rank deficiency of E) is chosen to minimize the euclidean norm of $\|AHy-(b-Ax_0)\|$ subject to $CHy \geq d-Cx_0$. Note that this is Problem P without the equality constraints of Eq. (1.2).

Let us now revert to the original notation. Consider first the case where the dimensions m_A and n satisfy $m_A \geq n$. You obtain a singular value decomposition $A = USV^T$ where $U^TU = I_n = VV^T = V^TV$ and $S = \text{diag}(s_1,\ldots,s_n)$, $s_1 \geq \cdots \geq s_n \geq 0$. Now for all x, $\|Ax-b\| = \|SV^Tx-g\|$ where

$U^Tb = g = [g_1,\ldots,g_{m_A}]^T$. Thus you have the equivalent problem of finding y to minimize $\|Sy-g\|$ subject to $CVy \geq d$, $(y=V^Tx)$.

If the diagonal matrix S was nonsingular we could let $z = Sy-\tilde{g}, \tilde{g} = [g_1,\ldots,g_n]^T$. Then the 'hard core' problem would be to minimize $\|z\|$ (or $\|z\|^2/2$, equivalently) subject to $CVS^{-1}z \geq d - CV\tilde{g}$. This is exactly Problem Q with proper relabeling of the variables. In case S is singular (say $s_r > 0$ and $s_{r+1} = \cdots = s_n = 0$) you will note that for every y, $\|Sy-g\|^2 = (s_1y_1-g_1)^2 +\cdots+ (s_ry_r-g_r)^2 + g_{r+1}^2 +\cdots+ g_{m_A}^2$.

Thus the minimum of $\|Sy-g\|^2$ subject to the constraints will occur at the same point as the minimum of $(s_1y_1-g_1)^2 +\cdots+ (s_ry_r-g_r)^2$ subject to these same constraints. Now the coordinates of y are transformed by setting $s_1y_1-g_1 = z_1,\ldots,s_ry_r-g_r = z_r$, $y_{r+1} = z_{r+1},\ldots,y_n = z_n$. If you put $\hat{S} = \operatorname{diag}(s_1,\ldots,s_r,1,\ldots,1)$ you have the equivalent problem of minimizing $(z_1^2 +\cdots+ z_r^2)/2$ subject to $CV\hat{S}^{-1}z \geq d-CV\hat{g}$, where $\hat{g} = [g_1,\ldots,g_r,0,\ldots,0]^T$. This is Problem Q again.

In the case where $m_A < n$ you construct orthonormal matrices $Q_{m_A \times m_A}$ and K_{nxn} so that $QAK = [R,0]$ where R is $m_A \times m_A$ and upper triangular. With $x = Ky$, we'll have the problem of minimizing $\|Ry_1-Qb\|$ subject to $CKy \geq d$. Here I've written $y = [\overbrace{y_1^T}^{m_A}, \overbrace{y_2^T}^{n-m_A}]^T$. A singular value decomposition is now obtained for R and substitutions analogous for the case $m_A \geq n$ are again performed to obtain a problem of the class Problem Q. I'll not elaborate further on the case $m_A < n$.

Now I'll discuss the important details of the computation of Problem P and Problem Q.

<u>Solving a least squares problem with constraints</u>

Procedure L1$(A,b,E,f,C,d;m_A,m_E,m_C,n;x,res)$.

The problem (<u>Problem P</u>) is to <u>solve</u>

$$Ax \cong b \text{ (least squares)}, \quad A_{m_A \times n}$$

<u>subject to</u>

$$Ex = f \qquad , \quad E_{m_E \times n}$$

<u>and</u>

$$Cx \geq d \qquad , \quad C_{m_C \times n}.$$

Here m_A, m_E, m_C and n are each positive integers.

<u>Removing linear equality constraints from a standard quadratic programming problem.</u>

Step	
1	Compute $x_0 = E^+ f$ for the matrix E of rank $r_E \leq \min(m_E,n)$. (See Hanson-Lawson [5] for an algorithm which effectively computes x_0.)
2	We will have x_0 as a numerically acceptable particular solution to $Ex=f$ provided $\|Ex_0-f\| \leq \epsilon[\|f\|+\|x_0\|\cdot\|E\|]$. If this condition is not satisfied, make an error indication and stop. (Take $\|E\|$ to be the maximum of row sums of absolute values of components of E.)

3 Else $b-Ax_0$ and $d-Cx_0$, respectively replace b and d in storage.

4 Let H denote the $n \times (n-r_E)$ matrix whose columns form an orthonormal basis for all solutions to $Ex=0$. The most general solution of $Ex=f$ is $x=x_0+Hy$, y an arbitrary $(n-r_E)$ vector. Then AH and CH replace A and C in storage. (See Hanson and Lawson [5] for an algorithm which does this in an efficient way.)

5 Execute the procedure $L2(A,b,C,d;\ m_A,m_C,n-r_E;y,res)$ to solve the reduced problem:

<u>Solve</u> $AHy \cong b-Ax_0$ (<u>least squares</u>)

<u>subject to</u> $CHy \geq d-Cx_0$.

6 The solution is computed as $x = x_0+Hy$, where $res = \|Ax-b\|$ is computed by the procedure L2.

<u>Reformulating a quadratic programming problem as a problem of the class 'Problem Q'</u>

Procedure L2 $(A,b,C,d;\ m_A,m_C,n;x,res)$

The problem here is to <u>solve</u>

$$Ax \cong b,\ A_{m_A \times n},\ (\underline{\text{least squares}})$$

<u>subject to</u>

$$Cx \geq d,\ C_{m_C \times n}.$$

This calculation can be done with the following steps.

Step

1 Have AD replace A in storage where D is chosen so that each non-zero column of A has euclidean length one.

2 If $m_A < n$ go to step 8.

3 Compute a singular value decomposition for A, $A=USV^T$, where $U^Tb = g = [g_1,\ldots,g_{m_A}]^T$ replaces b in storage, V replaces the first n rows and columns of A in storage, and $S = \mathrm{diag}(s_1,\ldots,s_{r_A},0,\ldots,0)$, with $s_1 \geq \cdots \geq s_{r_A} > 0$, $(r_A \leq n)$. (See Golub and Reinsch [4] for an efficient procedure for this.)

4 CV replaces C in storage. Put $C = [\overset{r_A}{\overline{C_1}},\ \overset{n-r_A}{\overline{c_2}}]$. Next $C_1\,\mathrm{diag}(s_1,\ldots,s_{r_A})^{-1}$ replaces C_1 in storage, and then $d-C_1[g_1,\ldots,g_{r_A}]^T$ replaces d in storage.

Step

5 We now have the problem:

<u>minimize</u> $(z_1^2+\cdots+z_{r_A}^2)^{\frac{1}{2}}$, $(r_A \leq n)$, <u>subject to</u>

$Cz \geq d$, $z = [z_1,\ldots,z_n]^T$.

This hard core problem is solved with the procedure L3(C,d; m_C,n,r_A; x,res)

6 Let $x = [\overbrace{x_1^T}^{r_A}, \overbrace{x_2^T}^{n-r_A}]^T$. Then $x_1+[g_1,\ldots,g_{r_A}]^T$ replaces x_1 in storage. Next $x_1\mathrm{diag}(s_1,\ldots,s_{r_A})^{-1}$ replaces x_1 in storage.

7 Finally Vx and Dx consecutively replace x in storage, and the procedure is completed for the case $m_A \geq n$ upon replacing res in storage by $(res^2+g_{r_A+1}^2+\cdots+g_{m_A}^2)^{\frac{1}{2}}$.

8 In case $m_A < n$, compute orthonormal matrices Q and K such that $QAK = [R,0]$, where R is upper triangular and $m_A \times m_A$. Then QA and Qb replace A and b in storage. Next AK and CK respectively replace A and C in storage. (It is important to note that the matrix K needs (essentially) just the last $n-m_A$ columns of A for storage of the data which describes it.) Now the $m_A \times m_A$ upper triangular matrix R has replaced the first m_A columns of A in storage.

9 We now have the problem of minimizing $\|Rx_1-b\|$ subject to $C_1x_1 + C_2x_2 \geq d$. Here $C = [\overbrace{C_1}^{m_A}, \overbrace{C_2}^{n-m_A}]$, and $x = [\overbrace{x_1^T}^{m_A}, \overbrace{x_2^T}^{n-m_A}]^T$. Now a singular value decomposition $R = USV^T$ is obtained. The equivalent problem is to minimize $\|Sz_1-U^Tb\|$ subject to $C_1Vz_1 + C_2x_2 \geq d$, $(z_1=V^Tx_1)$.

This allows us to replace C_1 and b in storage by C_1V and U^Tb respectively.

Step

9 (cont'd.) Let $U^T b = g = [g_1,\ldots,g_{m_A}]^T$.

10 Suppose $r_A \le m_A$ is the rank of A. Then $s_1 \ge \cdots \ge s_{r_A} > 0$ and $s_{r_A+1} = \cdots = s_{m_A} = 0$.

Repartition the matrix $C = [\overset{r_A}{\overline{C_1}}, \overset{n-r_A}{\overline{C_2}}]$.

11 Now $C_1 \mathrm{diag}(s_1,\ldots,s_{r_A})^{-1}$ replaces C_1 in storage, and then $d - C_1[g_1,\ldots,g_{r_A}]^T$ replaces d in storage.

12 We then have the problem of minimizing $(x_1^2+\cdots+x_{r_A}^2)/2$ subject to $Cx \ge d$. This 'hard core' problem is solved with the procedure L3(C,d; m_C,n,r_A; x,res).

13 Let $x = [\overset{r_A}{\overline{x_1^T}}, \overset{n-r_A}{\overline{x_2^T}}]^T$. Then $x_1+[g_1,\ldots,g_{r_A}]^T$ replaces x_1 in storage. Next $x_1 \mathrm{diag}(s_1,\ldots,s_{r_A})^{-1}$ replaces x_1 in storage.

14 Repartition $x = [\overset{m_A}{\overline{x_1^T}}, \overset{n-m_A}{\overline{x_2^T}}]^T$. Then Vx_1 replaces x_1 in storage.

15 Now Kx and Dx consecutively replace x in storage. The solution is computed upon replacing res in storage by $(res^2+g_{r_A+1}^2+\cdots+g_{m_A}^2)^{\frac{1}{2}}$

Remark

In Step 3 and Step 10 the (numerical) choice of the rank of the diagonal matrix S is often a nontrivial problem.

If $s_\ell/s_1 < \epsilon$ ($\ell=\min(m_A,n)$) then the rank should be no larger than the smallest integer r such that $s_{r+1}/s_1 < \epsilon$. Call this integer r_1; if $s_\ell/s_1 \geq \epsilon$ put $r_1 = \ell$.

Now if the least squares Problem P has had each row weighted by the reciprocal of the standard deviation of the error of the corresponding observation, then one should find the smallest integer, say r_2, such that $\|Ax-b\|^2 = \sum_{j=r_2+1}^{m_A} g_j^2 < m_A$. (This allows for an error no less than the magnitude of the standard deviation of the observation error to occur in each component of the residual vector Ax-b.) Finally set $r_A = \min(r_1,r_2)$.

<u>Hard core quadratic programs</u>: <u>minimize</u> $f(x) = \frac{1}{2}\sum_{i=1}^{r} x_i^2$ <u>subject to</u>

$Cx \geq d$, C_{mxn}.

Procedure L3 (C,d; m,n,r; x,res)

This is done with the following steps.

Step	
0	If m=0 then set x=0, and quit; if $0 \geq d$ then set x=0 and quit.
1	Else, normalize the m rows of C to each have euclidean length one. Multiply d by the diagonal matrix so defined. Here C and d can be modified in the same storage.
2	Separate the integers 1,...,m into two ordered disjoint groups, $\underline{\mathcal{E}}$ and $\underline{\mathcal{P}}$. The class $\underline{\mathcal{E}}$ will consist of all those integers for which the corresponding numbered constraint is violated or exactly satisfied. The class $\underline{\mathcal{P}}$ contains the rest. Numerically, $c_i^T x = d_i$ if $\lvert c_i^T x - d_i \rvert \leq \epsilon \left[\lVert x \rVert + \lvert d_i \rvert \right]$.
3	If there are no violated constraints go to Step 7 unless r=0 in which case quit.
4	Else place the rows from C and the components of d indexed by members of $\underline{\mathcal{E}}$ to a working array A and a working vector b.
5	Now b-Ax replaces b in storage.

Step

6 Compute $dx = A^+b$ (see Hanson and Lawson for a computational procedure [5]). Next $x+dx$ replaces x in storage. Then go to Step 2 unless more than $\max(m,n)$ iterations of Steps 2-6 have been executed. In the latter case quit; the constraints $Cx \geq d$ are contradictory.

<u>Remark</u>

The algorithm of Steps 2-6 was suggested by Rosen in [3]. No proof of convergence was given there.

7 Compute $\underline{\mathcal{E}}$ and $\underline{\mathcal{P}}$, and introduce a third class for the integers $1,\ldots,m$, say $\underline{U}$. The class $\underline{U}$ is disjoint from $\underline{\mathcal{P}}$. The class $\underline{U}$ has k members, $\underline{\mathcal{E}}$ has t members, and $\underline{\mathcal{P}}$ has $m-k-t$ members. Initially set $k=0$ or $\underline{U}=\emptyset$.

8 If x is not satisfying <u>some</u> constraint exactly, compute $t = \max_{1\leq i\leq m}(d_i/c_i^T x)$. Then tx replaces x in storage. Go to Step 7.

9 Else let $g=\text{grad}(f(x))=(x_1,\ldots,x_r,0,\ldots,0)^T$.

10 For $k > 0$ compute $y=A_k^+g$ with the procedure L4 (A,k,n,g).

Here $A_k^+g = R^{-1}d_1$, with $Q_k\ldots Q_1g = [\overbrace{d_1^T}^{k}, \overbrace{d_2^T}^{n-k}]$. The matrices $Q_1,\ldots,Q_k$ are orthonormal and R is $k\times k$ upper triangular and nonsingular.

11 Compute $\|A_ky-g\| = \|d_2\|$ and find q and c such that $c = y_q = \min_{i\in\underline{U}} y_i$. Set $c = 0$ if $\underline{U} = \emptyset$, (or $k=0$).

12 Compute $e = \epsilon[\|y\|+\|d_2\|]$.

13 If $c \leq -e$ go to Step 23.

Step

14 Else if $\|d_2\| \le \epsilon[\|x\|+\|y\|]$ then x is an optimum primal solution, y is a dual solution, and $[2f(x)]^{\frac{1}{2}}$ is the value of $(x_1{}^2+\cdots+x_r{}^2)^{\frac{1}{2}}$. We set res $= [2f(x)]^{\frac{1}{2}}$ and quit.

15 Else compute $v = g - A_k y$, (=g, if k=0), and compute p and an integer q such that $p = c_q^T v = \max c_i^T v$, $(i \in \underline{\mathcal{E}},\ i \notin \underline{U})$. (If $\underline{\mathcal{E}} = \underline{U}$ set p=0.)

16 If $p > \epsilon\|d_2\|$ go to Step 22.

17 Else if r=n or k=0 go to Step 20

18 Else we compute the unconstrained minimum of f(x) subject to the equality constraints that x remain on the linear flat of all solutions to $A_k^T x = d_k$. Here A_k is the matrix by the same name as that of Step 10 and d_k is the vector formed with the components of d corresponding to the $i \in \underline{U}$ in the same order as their appearance as rows A_k^T, (or columns of A_k).

The decomposition $[R^T, 0]^T = Q_k \cdots Q_1 A_k$ (already computed!) yields $[R^T, 0] = A_k^T Q_1 \ldots Q_k$. Now let

$$y = (\overbrace{y_1^T}^{k}, \overbrace{y_2^T}^{n-k})^T,\ Q = Q_1 \ldots Q_k, \text{ and } Q^T \tilde{v} = y.$$

We now have the (equivalent) least squares problem of minimizing the euclidean norm of $(I_r \oplus O_{n-r})(x - \tilde{v})$ subject to $A_k^T \tilde{v} = 0$. We also have the equation $A_k^T Q\, Q^T \tilde{v} = [R^T, 0] \cdot y = 0$. A particular solution to this equation is obtained with $y_1 = 0$. The most general solution to $A_k^T \tilde{v} = 0$ is given by $y = (y_1^T, y_2^T)^T$ where y_2 is arbitrary.

The substitution $\tilde{v} = Qy$ gives the rx(n-k) least squares problem $\hat{Q} y_2 = x_1$ for the (as yet arbitrary) segment y_2 of the vector y. Here x_1 is the first r components of the n vector x. The $r \times (n-k)$ matrix $\hat{Q}$ is the first r rows of the $n \times (n-k)$ matrix $Q_1 \ldots Q_k [0, I_{n-k}]^T$.

Operationally the data for $\hat{Q}$ can be generated in the last n-k columns of the nxn working array where the matrices A_k reside.

Finally the vector $\hat{x}$ which minimizes $f(x) = \sum_{j=1}^{r} x_j^2/2$ on the linear flat of all solutions to $A_k^T x = d_k$ is given by $x - \tilde{v}$, where $\tilde{v} = Q_1 \ldots Q_k y$.

19 Now compute the quantity $s = \max c_i^T \tilde{v}$, $(i \in \underline{\mathcal{P}}, i \notin \underline{U})$. Set $s=0$ if $\underline{\mathcal{P}} = \underline{U}$. If $s \le \epsilon \|\tilde{v}\|$ replace v in storage by $\tilde{v}$, set a=1, and go to Step 21.

20 Else compute the unconstrained minimum $f(x-av) = \sum_{j=1}^{r} (x_j - a v_j)^2/2$ for a by means of the formula $a =$

$$\sum_{j=1}^{r} x_j (v_j/\hat{v}) \Big/ \sum_{j=1}^{r} v_j (v_j/\hat{v}), \quad \hat{v} = \max_{1 \le j \le r} |v_j|.$$

21 Compute the smallest ratio $w = (c_i^T x - d_i)/c_i^T v$ over these $i \in \underline{\mathcal{P}}$ for which $c_i^T v \ge \epsilon \|v\|$. Put $w = a$ if $\underline{\mathcal{P}} = \emptyset$ or $c_i^T v \le \epsilon \|v\|$ for all $i \in \underline{\mathcal{P}}$. Then set $w := \min(w, a)$. Replace x in storage by $x - wv$. Go to Step 7.

22 Move q to $\underline{U}$. Recompute the pseudoinverse by adding a_q to A_k with procedure L5(A,k,n,a_q,P). Go to Step 9.

23 Move q from $\underline{U}$. Recompute the pseudoinverse by deleting a_q from A_k with procedure L6(A,k,n,a_q). Go to Step 9.

Remarks

In Steps 12 and 13 we say that all the dual coefficients are nonnegative (numerically) if the smallest of them exceeds $-\epsilon[\|y\|+\|d_2\|]$. This is based on the posterior estimate for errors in y caused by errors in the data, (see [5], Eq. (2.4.9)),

$$\|dy\| \le \frac{\|A_k^+\|}{1-\kappa\rho_{A_k}} \|dx\| + \|dA_k\| \left[\|y\|+\right.$$

$$\left.\frac{\|A_k^+\|}{1-\kappa\rho_{A_k}} \|d_2\|\right], \quad \rho_{A_k} = \|dA_k\|/\|A_k\|$$

We make the simplifying assumptions that $\|A_k^+\| = \|A_k\| = 1$ so that $\kappa = \|A_k^+\|\|A_k\| = 1$; $1-\kappa\rho_A \cong 1$, and $\|dA\| \cong \epsilon$. Since x is exact, $\|dx\|=0$. Combining these terms we have the (approximate) inequality $\|dy\| \le \epsilon[\|y\|+\|d_2\|]$. Thus the largest error in any component of y exceeds $-\epsilon[\|y\|+\|d_2\|]$. This is the worst case which would still have yielded a nonnegative dual solution had it not been for data errors in A_k. We allow always for this worst case to occur in our computation of a nonnegative solution of $A_k y=g$.

<u>Implicit pseudoinverse computation. Column deletion and insertion for use with gradient projection algorithms.</u>

Procedure L4 (A,k,n,g). Computing A_k^+g

We have $\begin{bmatrix} R \\ 0 \end{bmatrix} = Q_k \cdots Q_1 A_k$ where the Q_i are orthonormal and R is kxk and upper triangular.

Step	
1	If k=0, quit.
2	Else the product $Q_i \cdot g$ replaces g in storage, $(i=1,\ldots,k)$.
3	The product $R^{-1}[g_1,\ldots,g_k]^T$ replaces the first k components of g in storage. This product is computed by solving $Rz=g$ with z replacing the first k components of g in storage.

Procedure L5 (A,k,n,a_q). Inserting a column in the matrix A_k.

Step	
1	The vector a_q becomes the $k+1^{st}$ column of a working array A, whose first k columns are $Q_k \ldots Q_1 A_k$.
2	The matrix products $Q_i a_{k+1}$, $(i=1,\ldots,k)$, replace a_{k+1} in storage.
3	Compute an orthonormal transformation Q_{k+1}, $(i=1,\ldots,k)$ so that a_k is zero in components $k+2,\ldots,n$. (See [5])

Procedure L6 (A,k,n,a_q)

<u>Deleting a column in the matrix</u> A_k.

Step

1 The column vector a_q occupies column ℓ of the working array A. Delete a_q by restoring the original column vectors of A, $a_{\ell+1},\ldots,a_k$; have them replace the columns $a_\ell,\ldots,a_{k-1}$ of the array A in storage.

2 Have $Q_i a_j$ replace a_j in storage for $i=1,\ldots,\ell-1$, $j=\ell,\ldots,k-1$.

Now $Q_{\ell-1}\cdots Q_1 A_k = \begin{array}{cc} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} & \begin{array}{l} \}\ \ell-1 \\ \}\ n-\ell+1 \end{array} \\ \begin{array}{cc} \ell-1 & k-\ell \end{array} & \end{array}$ where R_{11} is upper triangular.

3 Continue the forward triangularization to obtain $Q_{k-1}\cdots Q_1[a_1,\ldots,a_{q-1},a_{q+1},\ldots,a_k] = [R^T,0]$ where R is upper triangular and $(k-1)x(k-1)$. Set $k: = k-1$, and quit.

The computational algorithms of Section 3 have been programmed in FORTRAN V (Univac 1108). Arrangements for acquisition of this code can be made by contacting the author.

4. REFORMULATING OTHER QUADRATIC PROGRAMMING PROBLEMS

Frequently a quadratic programming problem is given in the form

$$\underline{\text{minimize}}\ \tfrac{1}{2}x^T Bx + a^T x$$

subject to

$$Cx \geq d.$$

We assume here that explicit equality constraints have been removed. (This is done exactly as in Procedure L1). The matrix B is n x n, positive semi-definite and symmetric, while C is m x n.

To this end let

$$B = USV^T$$

be a singular value decomposition of B. Since B is symmetric and positive semidefinite, we may take U = V. We have $S = T \oplus 0_{n-r}$, where r is the rank of B, T consists of the positive eigenvalues of B,

$$b = (T^{-\frac{1}{2}} \oplus I_{n-r})Va = [\overbrace{b_1}^{r}, \overbrace{b_2}^{n-r}]^T$$

and

$$y = [\overbrace{y_1}^{r}, \overbrace{y_2}^{n-r}]^T.$$

Then with $y = (T^{\frac{1}{2}} \oplus I_{n-r})V^T x + [b_1, 0]^T$, we have

$$\tfrac{1}{2}x^T Bx + ax = \tfrac{1}{2}\|y_1\|^2 + b_2^T y_2 - \|b_1\|^2/2 .$$

The constraint

$$Cx \geq d$$

becomes

$$C'y \geq d',$$

where

$$C' = CV(T^{-\frac{1}{2}} \oplus I_{n-r})$$

and

$$d' = d + CV(T^{-\frac{1}{2}} \oplus I_{n-r})[b_1, 0]^T.$$

A trivial further simplification can be made by letting $\tilde{B}z_2 = y_2$ where $\tilde{B}$ is a diagonal matrix whose elements are the reciprocals of the components of b_2, if they are nonzero, and are ones otherwise. The vector y_2 can be permuted so that all of its leading components are ones. In this case the 'hard core' quadratic program becomes

<u>given</u> $0 \leq r \leq s \leq n$,

<u>minimize</u> $\varphi(x) = \frac{1}{2}(x_1^2 + \cdots + x_r^2) + (x_{r+1} + \cdots + x_s)$

<u>subject to m inequality constraints</u>.

$$Cx \geq d$$

This presents no difficulty; the gradient computation is changed in Step 9 of procedure L3 of Section 3:

$$g(x) = (\overbrace{x_1,\dots,x_r}^{r},\ \overbrace{1,\dots,1}^{s-r},\ \overbrace{0,\dots,0}^{n-s})^T$$

The computation of the unconstrained minimum along the linear flat of dual active constraints must be modified in an obvious way, too.

In case the matrix B is <u>positive</u> <u>definite</u> a Cholesky decomposition $B = F^TF$ can be obtained where F is a triangular nonsingular matrix. Now let $w = Fx + z$. The vector z is chosen so that $\frac{1}{2}x^TBx+a^Tx = \frac{1}{2}w^Tw$ + const. An equivalent problem is to minimize $\frac{1}{2}w^Tw = \|w\|^2/2$ subject to

$$CF^{-1}w \geq CF^{-1}z + d.$$

The execution of procedure L3 computes the solution to this problem.

5. AN APPLICATION: CURVE FITTING WITH CONSTRAINTS

In a great variety of applications there is a need to (least squares) fit data by curves which exhibit qualitative properties such as monotonicity and convexity over certain subintervals of the interval on which the data are given.

One well known way to construct such curves is to use cubic splines; the various constraints in the fitted curve become linear equality and inequality constraints on the coefficients which define the cubic spline. (See Amos and Slater, [6].)

More precisely, suppose m data triples are given, say $\{(x_i,y_i,\sigma_i)\}$. The data points x_i are the independent variables, the points y_i are the observations, and the σ_i are the standard deviations of the observations.

The points x_i (which we assume are increasing with i) lie in an interval $[a,b]$ with $a \le x_1$ and $x_m \le b$.

We assume that n+2 fixed breakpoints $b_0 \le b_1 \le \ldots \le b_n \le b_{n+1}$, ($b_0 = a$, $b_{n+1} = b$) are given.

For purposes of scaling, the interval $[a,b]$ should be changed to $[-1,+1]$ by the transformation $t = (2x-s)/d$, $(d=b-a,\ s=a+b)$. This transformation is applied to the breakpoints b_i, $(i=0,\ldots,n+1)$, together with the data points x_i, $(i=1,\ldots,m)$. For convenience we now revert to the original notation.

We define a twice differentiable function $p(x)$ on $[-1,1]$ by cubics

$$p_i(x) = c_{i1}(x-b_{i-1})^3+c_{i2}(x-b_{i-1})^2+c_{i3}(x-b_{i-1})+c_{i4} \text{ for}$$

$$b_{i-1} \le x < b_i,\ (i=1,\ldots,n+1). \tag{5.1}$$

The function p so defined will be twice continuously differentiable provided

$$p_i(b_i) = p_{i+1}(b_i)$$

$$p_i'(b_i) = p_{i+1}'(b_i), \; (' = \frac{d}{dx}),$$

and

$$p_i''(b_i) = p_{i+1}''(b_i) \tag{5.2}$$

at the interior breakpoints (or <u>knots</u>) b_i, $i=1,\ldots,n$.

The fitted curve and its first two derivatives will interpolate certain values if additional equality constraints

$$p^{(i)}(\hat{x}_j) = y_j^{(i)}, (i=0,1,2), \tag{5.3}$$

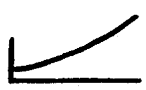
are specified.

These conditions allow you to force the fitted curve to pass through certain points, have a local minimum or maximum where desired, and force points of inflexion to occur at other given points.

Since $p''(x)$ is a piecewise linear function, its sign throughout any subinterval $[x_1', x_2']$ can be fixed (nonnegative) by having $p''(x_1') \geq 0$ and $p''(x_2') \geq 0$. Analogous remarks hold when fixing the sign of $p''(x)$ to be nonpositive on $[x_1', x_2']$.

These remarks show that the convexity (downward or upward) of p can be designated on any subinterval of $[-1,+1]$ since $p'' \geq 0$ implies p is convex on that subinterval.

More generally consider the four <u>basic</u> <u>shapes</u> which can be 'pieced together' to form a large class of functions p(x):

 1. p convex downward and increasing on $[x_1', x_2']$

 2. p convex downward and decreasing on $[x_1', x_2']$

 3. p convex upward and increasing on $[x_1', x_2']$

 4. p convex upward and decreasing on $[x_1', x_2']$ (5.4)

These four basic shapes can be achieved by the respective constraints

1. $p''(x_1') \geq 0,\ p''(x_2') \geq 0,\ p'(x_1') \geq 0$

2. $p''(x_1') \geq 0,\ p''(x_2') \geq 0,\ p'(x_2') \leq 0$

3. $p''(x_1') \leq 0,\ p''(x_2') \leq 0,\ p'(x_2') \geq 0$

4. $p''(x_1') \leq 0,\ p''(x_2' \leq 0,\ p'(x_1') \leq 0$ (5.5)

For notational convenience, I'll define the 4(n+1) - dimensional vector c of the coefficients,

$$c = [c_{11}, c_{12}, c_{13}, c_{14}, c_{21}, c_{22}, c_{23}, c_{24}, \ldots, c_{n+1,1},$$

$$c_{n+1,2}, c_{n+1,3}, c_{n+1,4}]^T \tag{5.6}$$

Generally speaking the desired form of the curve gives rise to

1. The equality constraints for continuity. (Matrix E)
2. The equality constraints for interpolation of the curve and its derivatives through specified values. (Matrix E)
3. The inequality constraints for achieving basic shapes. (Matrix C)
4. The inequality constraints for bounding the curve and its derivatives. (Matrix C)

Many other desired properties of the fitted curve can be posed as linear equality and inequality constraints on the vector c. I will not elaborate on these.

The fitted curve is to least squares fit the m data points with respect to the weighting matrix $W = \operatorname{diag}(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_m})$.

Thus we are led to the following quadratic programming problem for the vector c:

$$\begin{array}{ll} \underline{\text{minimize}} & \|W(Ac-d)\| \\ \underline{\text{subject to}} & Ec = f \\ \underline{\text{and}} & \\ & Cc \geq e. \end{array} \tag{5.8}$$

The matrices A, E and C, and the vectors d, f and e are defined as follows:

Let m_i, $i=1,\ldots,n+1$, be the number of data points in the subinterval $[b_{i-1},b_i]$. Then A is an $m \times 4(n+1)$ block diagonal matrix

$$A = \begin{bmatrix} A_1 & & & 0 \\ & A_2 & & \\ & & \ddots & \\ 0 & & & A_{n+1} \end{bmatrix} \tag{5.9}$$

where each A_i is an $m_i \times 4$ matrix

$$A_i = \begin{bmatrix} (x_k-b_{i-1})^3, & (x_k-b_{i-1})^2, & (x_k-b_{i-1}), & 1 \\ & \vdots & & \\ (x_{k+m_i-1}-b_{i-1})^3, & (x_{k+m_i-1}-b_{i-1})^2, & (x_{k+m_i-1}-b_{i-1}), & 1 \end{bmatrix} \tag{5.10}$$

$$i=1,\ldots,n+1$$

Here the x_j indexed $k,\ldots,k+m_i-1$ denote just those x_j such that $x_{k-1} < b_{i-1} \leq x_k \leq \ldots \leq x_{k+m_i-1} < b_i$.

That $3n \times 4(n+1)$ portion of the matrix E, $E^{(1)}$ which determines the $\mathcal{C}^2$ continuity of the curve has the block diagonal form

$$E^{(1)} = \begin{pmatrix} E_1 & & 0 \\ & \ddots & \\ 0 & & E_n \end{pmatrix} \tag{5.11}$$

where each E_i is a 3×8 matrix

$$E_i = \begin{bmatrix} d_i^3, d_i^2, d_i, 1, 0, 0, 0, -1 \\ 3d_i^2, 2d_i, 1, 0, 0, 0, -1, 0 \\ 6d_i, 2, 0, 0, 0, -2, 0, 0 \end{bmatrix}, \tag{5.12}$$

$$d_i = b_i - b_{i-1}$$

That portion of the matrix E which specifies that the curve or its derivatives are to interpolate values is given by the appropriate $4(n+1)$ row vectors

$$E_i^{(2)} = [\overbrace{0,\ldots,0}^{4j},d_i^3,d_i^2,d_i,0,\ldots,0] \tag{5.13}$$

$(p(x_i) = y_i^{(0)})$,

$$b_j \le x_i < b_{j+1},\ c_i = x_i - b_j$$

$$F_i^{(2)} = [\overbrace{0,\ldots,0}^{4j},3d_i^2,2d_i,1,0,0,\ldots,0] \tag{5.14}$$

$(p'(x_i) = y_i^{(1)})$

$$b_j \le x_i \le b_{j+1},\ d_i = x_i - b_j$$

and

$$G_i^{(2)} = [\overbrace{0,\ldots,0}^{4j},6(x_i-b_j),2,0,0,0,\ldots,0] \tag{5.15}$$

$(p''(x) = y_i^{(2)})$

$$b_j < x_i \le b_{j+1},\ d_i = x_i - b_j$$

The matrix C will consist of several (say ℓ) rows each with the form of one of the rows $E_i^{(2)}$, $F_i^{(2)}$ and $G_i^{(2)}$ of Eqs. (5.13)-(5.15). Inequality constraints $p(x_i) \le e_i$ will change the sign of the four nonzero entries of $E_i^{(3)}$. Analogous remarks hold when forming rows corresponding to $p'(x_i) \le e_i$ or $p''(x_i) \le e_i$.

The vectors d, f and e are defined by

$$d = [y_1,\ldots,y_n]^T \quad (y_i = \text{observations}),$$

$$f = [\overbrace{0,\ldots,0}^{3n},y_1^{(0)},\ldots,y_k^{(2)}]^T, \quad (y_i^{(j)} = \text{interpolated values for derivatives}),$$

and

$$e = [e_1,\ldots e_\ell]^T \quad (e_i = \text{bound for the curve or its derivatives; if this was an upper bound, change the sign of } e_i) \tag{5.16}$$

Finally, the vector of coefficients, c, can be obtained by executing the procedure L1(WA,Wd,E,f,C,e,m,3n+k,ℓ,4n+4;c,res), ($k \ge 0$, $\ell \ge 0$), which was given in Section 3 of this paper. If the required constraints desired for the fitted curve can be realized, a vector c will always be given by procedure L1. This can be a welcome component of interactive lightpen-data fitting software systems.

The evaluation of the fitted curve at any point $a \le x \le b$ is done by computing

1. $t = (2x-s)/d$; $(d=b-a, s=a+b)$

2. Find i such that $b_{i-1} \leq t < b_i$, $(1 \leq i \leq n+1)$.

3. Compute $y = t-b_{i-1}$

4. Compute $p(x) = ((c_{i1}y+c_{i2})\ y+c_{i3})y+c_{i4}$

Compute $p'(x) = [(3c_{i1}y+2c_{i2})y+c_{i3}][2/d]$

Compute $p''(x) = [6c_{i1}y+2c_{i2}]\{[2/d]^2\}$.

REFERENCES

[1] Saaty, T. L. and Bram, J., Nonlinear Mathematics, McGraw-Hill, New York, (1964).

[2] Golub, G. and Saunders, M. A., Linear Least Squares and Quadratic Programming, Tech. Report CS134, Computer Sciences Dept., Stanford University, California, (1969).

[3] Rosen, J. B., The Gradient Projection Method, Part I: Linear Constraints, J. Soc. Ind. and Appl. Math., Vol. 8, No. 1, Pp. 181-217, March (1960).

[4] Golub, G. H. and Reinsch, C., Singular Value Decomposition and Least Squares Solution, Tech. Report No. CS131, Computer Sciences Dept., Stanford University, California, (1969).

[5] Hanson, R. J. and Lawson, C. L., Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems, Math. of Comp., Vol. 23, No. 108, Pp. 787-812, October (1969).

[6] Amos, D. E. and Slater, M. L., Polynomial and Spline Approximation by Quadratic Programming, Comm. of the ACM, Vol. 12, No. 7, Pp. 379-381, July (1969).